Evaluating Power Monitoring Capabilities on IBM Blue Gene/P and Blue Gene/Q

Kazutomo Yoshii, Kamil Iskra, Rinku Gupta, Pete Beckman, Venkatram Vishwanath, Chenjie Yu, Susan Coghlan

Argonne National Laboratory
Mathematics and Computer Science Division
9700 South Cass Avenue, Argonne, IL 60439, USA

***Abstract*—As we continue our quest toward exascale computing, power consumption is becoming a critical factor, along with resiliency and concurrency. Although power requirements of individual system components (e.g., processor, memory) are taken into consideration by vendors during the design phase, actual power consumption of a complete system is an insufficiently studied research area. Estimating the power consumption of a large-scale system is a nontrivial task because of the number of components involved and also because power requirements are affected by the (unpredictable) workloads. What is needed is a power monitoring infrastructure that can provide timely and accurate feedback to system developers and application writers so they can optimize the use of this precious resource.**

In this paper, we first summarize our prior power-related experiences and results on Blue Gene/P. Then we outline the new power measurement capabilities of the system on IBM Blue Gene/Q, currently the most energy-efficient platform on the Green500 list. We describe the important characteristics of the power measurement capabilities and the challenges they present. We explain how we successfully implemented our power-profiling code and demonstrated it on Argonne's early-access Blue Gene/Q system. Using the profiling code, we characterized power consumption of primitive operations. In preparation for profiling power consumption of real-world applications, we evaluated the accuracy of the power measurement capabilities for short-duration activities.

I. Introduction

The field of supercomputing has grown by leaps and bounds since its emergence 50 years ago, and it is expected to enter the exascale era within a decade. In order to meet the exascale goals, today's top supercomputers will need to scale by two orders of magnitude, at the same time increasing their power consumption by only an integer factor, to no more than 20 MW. Power consumption of both individual nodes and the overall HPC system is thus a critical issue to address [1].

Most of the improvement required will need to take place on the hardware side. CPU designers have recognized that for some time now, reducing clock frequencies, choosing simpler, more power efficient core designs, and employing dynamic voltage and frequency scaling. Software will also have a role to play, however, both in ensuring that power is not wasted and in dynamically managing power consumption across the system.

Yet, as the saying goes, *You can't manage what you don't measure*. Most performance studies of large-scale HPC systems and their workloads (including parallel applications, communication libraries, and system software) have focused primarily on flops, bandwidth, and latency. Few concrete studies exist that focus on studying and quantifying power and energy consumption at the hardware and software level. Until recently, system vendors have had little incentive to expose extensive system and component-level power information to users. Consequently, the power-monitoring methodology is lacking and the measuring capabilities in today's computer systems are limited or missing. Petascale (and future exascale) systems consisting of hundreds of thousands of nodes drawing megawatts of electrical power mandate a need for new, systemwide methodologies and procedures for accurate and real-time power monitoring.

Currently, however, we are forced to rely on existing monitoring capabilities provided by vendors. Large-scale systems have built-in environmental monitoring capabilities designed primarily to help identify and eliminate insufficient cooling (through the use of temperature sensors) and inadequate distribution of electrical power (through voltage and current sensors). One of our main goals is to understand what power-monitoring capabilities are currently made available and how they fare with respect to spatial and temporal resolution, accuracy, latency, and other characteristics; how much useful information is provided by these power monitoring capabilities; and what level of information can actually be exposed to end users.

After outlining related work in Section II and IBM Blue Gene architecture in Section III, we present experiences and results from our early explorations on Blue Gene/P system in Section IV. Subsequently, in Section V, we provide an overview of the new, improved power measurement capabilities of Blue Gene/Q and the challenges they present. We also present the details of our power-profiling implementation and the measured power characteristics on primitive operations. In preparation for profiling the power consumption of real-world applications, we evaluate the accuracy of the power measurement capabilities for short-time activities. Section VI provides conclusions and future directions.

II. Related Work

Past research in the field of power usage, monitoring, and management has focused on different goals at various times.

From the system-level perspective, power consumption has increasingly been recognized as a limiting factor in large data centers and supercomputer facilities. Running and cooling large computing systems come with significant cost attached [2], [3], [4], [5], [6].

Traditionally, power-related studies have been conducted with a goal of reducing energy consumption, minimizing operating machine cooling costs and thus the total cost of ownership. These studies focused on exploring the possibility of integrating available information with other system software, such as runtime systems and the job scheduler, for optimal scheduling with reduced costs. Research in power-aware scheduling has been vast and diverse [7], [8], [9], [10], [11], [12], [13].

Garcia et al. [14] developed an instruction-level energy consumption model for many-core architectures and demonstrated its accuracy by experimenting on an IBM Cyclops-64 chip.

Feng et al. [15] developed a power/energy profiling framework for HPC cluster systems. They measure power consumption by tapping digital multimeters into DC lines. They studied the power-performance efficiency of the NAS parallel benchmarks on a 32-node cluster.

Alam el al. [16] evaluated IBM Blue Gene/P (BGP), comparing the Cray XT4 in various aspects of performance, including performance per watt. They also compared power consumption of HPC applications. They concluded that BGP showed better performance-per-watt for certain computational kernels while it had less power advantage on some scientific workloads such as the Parallel Ocean Program.

Hennecke et al. [] presented an overview of the power measurement capabilities of BGP. They measured power consumption on HPC applications and presented the integration of power and energy. No in-depth analysis of the accuracy of power measurement was conducted in that study.

Still working on this section..

III. Overview of the IBM Blue Gene supercomputers

The IBM Blue Gene supercomputers have been at the forefront of high-performance computing for several years. Blue Gene/L (BGL) [17]—the first generation—was released in 2004 and topped the world wide Top 500 supercomputer list [18] for three years. Along with scalability, IBM designed BGL with power efficiency in mind. BGL employed 32-bit PowerPC 440 processors running at 700 MHz, which were less powerful, but also considerably less power-hungry, than processors used in HPC clusters then. The core dissipated only 2.5 mW/MHz with 1.8 V input voltage (an estimate) [19], which yielded 3.5 W per BGL dual processor. In comparison, the thermal design power of a dual-core Intel Xeon processor in 2004 was more than 100 W [20].

In this paper, we focus on the Blue Gene/P [21], released in 2007, and the Blue Gene/Q (BGQ), which was released in 2012. Table I provides a brief overview of the architecture of BGP and BGQ. BGP is basically an improved version of BGL, doubling the core count, memory capacity, and bandwidth; slightly increasing core frequency; and employing a coherent cache. BGQ is the third generation of the IBM Blue Gene massively parallel supercomputer series. Its node cards are water-cooled whereas the previous generations were air-cooled. BGQ made a big leap in processor and interconnect technologies. The peak performance is 209 TF per rack, and the power efficiency is approximately 2.1 GFlops/W without optical interconnect.

Packaging and system management are also important design points of the Blue Gene architecture [21], [22]. Every rack of the Blue Gene system is organized in a hierarchical manner. A rack consists of two mid-planes, eight link cards, and two service cards. A mid-plane contains 16 node boards. Each node board holds 32 compute cards, for a total of 1,024 nodes per rack. For the BGP, each compute card has a single PowerPC 450 processor, with four cores operating at 850 MHz each. For the BGQ, each compute card contains a single 18-core PowerPC A2 processor [23] (16 cores for applications, one core for system software, and one core inactive) with four hardware threads per core, and DDR3 memory. BGP thus has 4,096 cores per rack, and BGQ has 16,384.

TABLE I
Blue Gene Architecture

	Blue Gene/P	**Blue Gene/Q**
Processor core	PowerPC 450 (32-bit)	PowerPC A2 (64-bit)
Speed	850 MHz	1600 MHz
# of cores	4	16 (+1 for system)
# of HW threads	1	4
Peak per node	13.6 GFlops	205 GFlops
L1 cache	32 KB D + 32 KB I	16 KB D + 16 KB I
Shared cache	8 MB (L3)	32 MB (L2)
Memory	2 or 4 GB	16 GB
Memory bandwidth	13.6 GB/s	42.6 GB/s
Power efficiency	357 MF/W	2 GF/W
Interconnect	3D torus	5D torus
Cooling	air cooling	water cooling

The Blue Gene systems have environmental monitoring capabilities for system health check and diagnostics. These capabilities periodically sample and gather environmental data from various sensors and store this collected information in a DB2 database (which can be read by privileged users) with time-stamp and location information. Sensors are present in locations such as service cards, node boards, bulk power modules and cooling system boards; and monitor various physical attributes such as temperature, coolant flow and pressure, fan speed, voltage and current. Sensor data are collected in a relatively long polling interval (5 minutes). While a shorter interval would be preferable, decreasing it stresses the database and, given the data volumes involved, can even exceed the database server's processing capacity. The default polling interval is 30 minutes for service cards and 5 minutes for node cards.

The BG database stores power consumption information (in watts) for the following components that are of particular interest to us in this paper:

- *Bulk*: AC/DC converter on a rack
- *Node board*: DC/DC converter on a node board

- *Link card*: DC/DC converter on a link card
- *Service card*: DC/DC converter on a service card

IV. Early Power Experiments on Blue Gene/P

Our early power experiments on BGP focused on evaluating the capabilities of the power monitoring while stressing different parts of the system. In particular, we focused on the Double-Hammer FPU unit (DFP), L1 cache, L3 cache, main memory, network link usage, and overall system (note that in BGP, the L2 cache is just a prefetching buffer, so we did not study it). In addition, we studied power consumption during two system states: (1) *Empty*—when the BGP nodes are not allocated to any user and are thus not booted and (2) *Sleep*—when the system is booted but idling.

Our studies were conducted on the Argonne National Laboratory's "Intrepid" [24] Blue Gene/P system, one of the largest BGP installations in the world. Intrepid has 40 compute racks with a total of 163,840 cores and a peak performance of 557 TFlops.

A. Microbenchmarks and Methodology

As a part of our study, we created microbenchmarks to stress Double Hammer FPU (DFP), L1 cache, L3 cache, and main memory.

DFP benchmark: This is a floating-point loop benchmark, which is designed to stress solely the FPU. The core of this benchmark is written in assembly language and repeats a series of the Double-Hammer FPU instructions. It is optimized to achieve the BGP peak performance, which is 3.4 Gflops per core.

Cache/Memory benchmark: The L1 cache, L3 cache, and main memory microbenchmarks share the same stress code, but we vary the buffer size depending on the target we are testing. The code performs a random walk over a pre-initialized memory buffer. The buffer size we used for the L1 and L3 cache studies was 32 KB and 8 MB, respectively. The buffer size for the main memory benchmark was 256 MB.

Network and system-level benchmarks: Our benchmark for network stressing was based on all-to-all communication using the MPI "all-to-all" [25] routines. For our system-level study, we used the "QCD" application, which is a lattice quantum chromodynamics code and is known to be an aggressive stress code for supercomputer systems.

Studying system states: In the Empty system state, most of the components are expected to be in low power mode. We studied this state since it occurs regularly when the job scheduler cannot make full use of the system because of scheduling conflicts between jobs. In the Sleep system state, the processes were suspended inside the `usleep()` system call.

Power measurement on BGP is coarse grained in both time and space. In order to obtain reliable measurements, constant and static workloads have to be run for a period at least twice as long as the polling interval (which can be set between 60 and 1800 seconds). We chose to run each experiment for an hour. To compare the bulk rack-level system power consumption with the power consumption of a specific system component, we ran each workload/experiment on an entire rack (i.e., 4,096 cores). As mentioned, the database stores power information for various components periodically. We used this database and mapped our experiment launch and execution time to derive power consumption for various components during the experiments.

B. Understanding Results

Link and Service cards: Table II presents the power consumption of link cards and service cards for the Empty and Sleep states and the DFP, L1, L3, Memory, AlltoAll, and QCD workloads. Note that one rack of BGP has two service cards and eight link cards and that the results presented are cumulative. As can be observed, the differences between power consumption levels of these components under different workloads are fairly small, with a total of around 800–850 watt per rack.

TABLE II
Link card and Service card power consumption (watts)

Card	Empty	Sleep	DFP	L1	L3	Memory	AllToAll	QCD
Link	341	343	342	371	369	371	371	342
Service	482	486	486	505	487	485	505	564

BGP components: Fans, Processors: A BGP rack has 60 fans. Each fan consumes up to 26 W, which yields 1.56 kW per rack. As the fans themselves do not have any DC/DC converter modules and draw power directly from rack AC/DC output, measuring their power consumption is difficult.

Each BGP compute node contains a quad-core PowerPC 450 CPU. Estimating from the PowerPC 440 core specification [19], each core runs at 2.5 mW/MHz, which at 850 MHz amounts to 8704 W for the whole BGP rack.

TABLE III
Bulk and Node board power consumption (kW)

	Empty	Sleep	DFP	L1	L3	Memory	AllToAll	QCD
Rack	8.7	21.4	23.3	22.8	22.9	30.4	22.8	28.4
Node boards	3.0	14.9	16.1	16.1	16.1	22.6	16.3	20.1

Node Boards: Table III presents the bulk rack-level power consumption from the line cord and the total power consumption of the node boards. Each rack has 32 boards; we present the cumulative power for all of them.

As can be observed, doing heavy computations using the *DFP* workload, with very little DRAM access, draws 23.3 kW per rack (16.1 kW from node boards). However, CPU power consumption does not show big variations when different parts are stressed: walking through L1 and L3 gives numbers very close to those of DFP. Even idling (*Sleep*) results in a drop of only about 1 kW per rack.

On the other hand, power consumption increases up to 30.4 kW (22.6 kW from node boards) when DRAM gets stressed (*Memory*). This clearly shows that memory operations are expensive in terms of power consumption compared to computational instructions on the processor: about 8 kW or 42% difference when there is heavy traffic to and from memory. Lattice QCD is heavy on both computation and memory access, which puts it close to the high end of power consumption, as can be seen from the table.

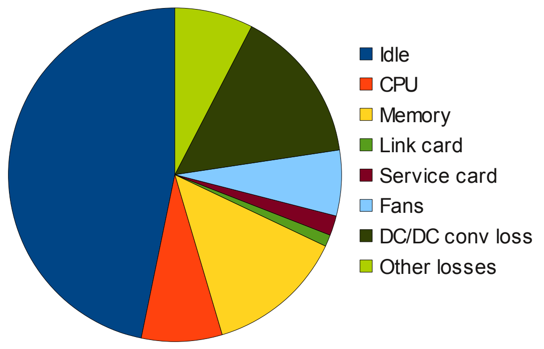

Fig. 1. BGP power distribution for QCD running on a single rack

Based on this analysis, we break down the power distribution for a single BGP rack when running the lattice QCD application and demonstrate it in Figure 1. The *Idle* value comes from a separate *Sleep* run; the *CPU* value is the extra CPU power draw estimated from the difference between *DFP* and *Sleep* workloads; the rest of node board power draw is attributed to *Memory*; the *Link card*, *Service card*, and *Fans* are assumed to be constant; *Other losses* contain the aggregated estimation error. We expect other applications to have similar power distributions.

In summary, we found that it is possible to extract interesting power consumption data from the BGP database. The practical value of that data, however, is severely limited by the long polling interval of the stored power data. Also, the data stored for compute node boards is cumulative—our attempts to separate CPU an memory consumption are full of assumptions that are difficult to verify.

V. Power Experiments on Blue Gene/Q

On BGQ, IBM provides new interfaces in the form of an *Environmental Monitoring (EMON) API* that allows one to access power consumption data from code running on compute nodes, with a relatively short response time.

Our experiences with BGQ are based on Argonne National Laboratory's one-rack BGQ system, which in its current form is a early-access system.

In this section, we first describe the details of the node board architecture. We then outline our power profiling code and discuss the results obtained with various workloads. We also discuss the accuracy of the power measurement capabilities for short-duration activities.

A. Node Board Architecture

A BGQ rack has 32 node boards, each of which hold 32 compute cards. Figure 2 shows a representation of a node board. Each node board had two direct-current assemblies (DCA). The DCA converts the 50 V DC input voltage to intermediate voltages. Each node board also has several voltage transformation modules (VTMs), which transform the intermediate voltage output from the DCA to lower voltages (based on a specific ratio) intended for different parts of the node board.

The board has several power domains, but power can currently only be measured in the seven power domains (listed in Table IV). The DCA, internally, has a microcontroller that periodically reads input current (I_{in}) and output voltage (V_{out}) from the VTMs for these seven different domains for each DCA. Instantaneous power (W) is calculated as $V_{\text{out}} \times I_{\text{in}} \times$ *ratio*.

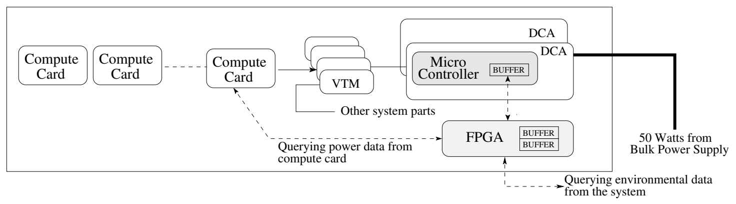
Compute Card
Compute Card
Compute Card
VTM
Other system parts
DCA
DCA
Micro Controller
BUFFER
FPGA
BUFFER
BUFFER
Querying power data from compute card
50 Watts from Bulk Power Supply
Querying environmental data from the system

Fig. 2. BGQ node board

TABLE IV
Node Board Power Domains

Voltage	Description
0.8	BGQ compute chip (BQC) and L2 (EDRAM)
1.4	Main memory (DDR3)
2.5	Optical module power related
3.3	Optical module power related
1.5	BQC and Link chip for torus links
0.9	BQC array (on-chip memory cells)
1.0	Link chip core power

Direct access to the internal microcontroller on the DCA is limited because internal buses and protocols on the board are proprietary. We can, however, obtain power and current information through an FPGA available on the node board. The FPGA periodically queries and obtains environmental data from the entire system. In addition, the FPGA frequently reads voltage and current from the DCA microcontroller's buffer for each of the power domains. It performs these buffer reads in a round robin manner. This requires 28 read accesses for the two DCAs. The read interval for the FPGA is currently set to 20ms; thus 560ms are needed to read all domains' voltage and current, and 560ms is therefore the sampling period. The FPGA stores this voltage and current information in a buffer that can hold 14 generations of it.

The lightweight operating system (called CNK) running on the compute nodes can query any generation of power data from the FPGA via an internal bus. The CNK provides a system call that can return raw data (voltage and current) of all domains; however, it currently returns only the oldest generation of power data (i.e. oldest of the 14 generation of power data; each of which were based on a 560ms sampling period), which is approximately 6-7 seconds old. This data may get even older if it overlapped with the BGQ control system accessing the FPGA at the same time. The actual age of the returned power data cannot be estimated currently. The system call used to get this information is called as the "the EMON system call." The latency of the system call itself is approximately 800 μs, however.

As mentioned earlier, IBM provides a new set of user-level interfaces called "the EMON APIs" for obtaining environment information. The EMON API, internally, relies on the "EMON system call", thus inheriting its limitations. In addition, the EMON API only returns the *total* power consumption of the oldest generation. Since the underlying power measurement infrastructure does not measure all domains at the same instant, the total power consumption the EMON API returns may be inconsistent in some cases. For example: If a workload begins stressing both the core and the memory at the same time, we may not see an increased power consumption on both core and memory within the same generation of results, as there is a time-gap between when the core and the memory power was measured.

We note that there is an on-going research activity in IBM to improve the power systems, so the attributes such as sampling interval on the BGQ power system may change in the future.

B. Implementation

The provided EMON API is currently insufficient for our purposes because it returns only the total power consumption of all domains and has no profiling functionality. Instead, we designed a "power profiler code" to use the EMON system call directly to allow us to read the 14 individual voltage and current data points. The disadvantage of using the system call directly, though, is that it forces us to maintain our own voltage and current conversion code.

Our profiler code is designed to run one thread on each node board. The thread periodically invokes the system call with an interval lower than the FPGA interval (i.e., 560ms), records the voltage and current on all the domains along with a timestamp, and dumps recorded data to files. While the power data we obtain from the EMON API is instantaneous power (watt), profiling power consumption also allows us to estimate integral power, or energy (i.e., watt-hours or joules). Later in this section, we discuss the accuracy of estimated energy from sampled power data.

An interesting implementation detail in the profiler code is the choice of the compute node to query the power interface and the placement of the power consumption profiling thread. Note that BGQ has one power monitoring unit per node board, and thus we need only one profiling thread per board. Since every node board has 32 compute nodes, we choose a compute node by directly decoding the node board and node card identifiers from the universal component identifier (UCI) available in the BGQ "personality" data structure.

The lightweight OS (CNK) running on the compute nodes allows us to over commit multiple logical threads on top of a hardware thread but it does not support a time-quantum-driven preemption. Hence, the profiling thread cannot call the system call to get power data while an application thread is busy performing computations or communication. We currently assign the profiling thread to the last hardware thread available to applications on the chosen node. The BGQ currently supports 64 hardware threads on each node, thus this approach will not work if an application uses all the 64 hardware threads. In practice, however, using all hardware threads is not necessarily beneficial for applications and may even cause performance degradation. However, in the interest of profiling applications that use all 64 threads, we are investigating the use of the 17th core, reserved for system software, to run our profiling thread.

C. Experimental Setup

We have successfully implemented our power profiling code that periodically samples all domains and records timestamped power data for analysis. In our experimental setup, the minimum partition size (due to cabling and control system limitations) that allows us to measure the power consumption is 128 nodes, which spans four node boards.

In this paper, we have focused on understanding the power consumption at a node board level for the FPUs and the memory subsystem. Additionally, we study some power consumption aspects of the communication subsystem by using MPI benchmarks.

To understand the power consumption on the FPU and memory, we designed a stress suite for many-core processors called manycore-heater. The suite spawns OpenMP threads and starts specified stress on every thread; it also uses MPI for internode synchronization and reporting aggregated bandwidth and flops from internal counters. A stress pattern workload consists of a series of dummy double-precision multiply-add operations, which allows us to understand how each component consumes power on a primitive operation. The start and end time of each stress test is measured for comparison with sampled power data.

While workloads from real-world applications are usually fixed work quanta (FWQ), our artificially generated stresses are fixed time quanta (FTQ): each stress pattern is repeated for a specified time duration. Comparing FWQ workloads in energy (joules or watt-hours) is practical. Since we use FTQ workloads, we measure our data in flops per watt or bandwidth per watt (MB/s/W), which can be written as operations per joule (the inverse of which is joules per operation).

D. Understanding Power Measurement Results

Figure 3 is a time series power-related graph obtained by running the profiling code with an artificial stress pattern on all 16 cores and 128 nodes. The pattern repeats a cycle of i seconds main memory stress workload and i seconds sleep with $i = 1, 2$, and 3. As described earlier, the system call returns approximately 7-seconds-old power data. The power profile code also captures the power activities before the profiling thread starts. This data may be useful in order to observe power consumption of the initialization phase (e.g., operating system startup) of the compute nodes.

FPU power consumption: BGQ A2 processor core has a SIMD quad-vector floating-point(FP) unit, or QPU, (theoretically) delivering 8 FP ops per cycle. QPU executes both

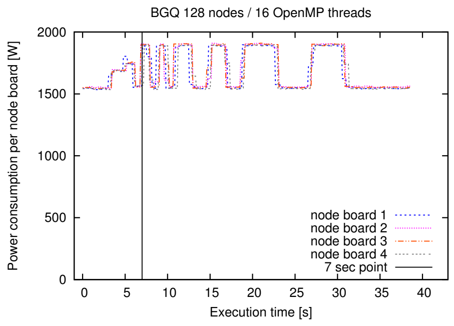

Fig. 3. Time series of aggregated power consumption per 128-node

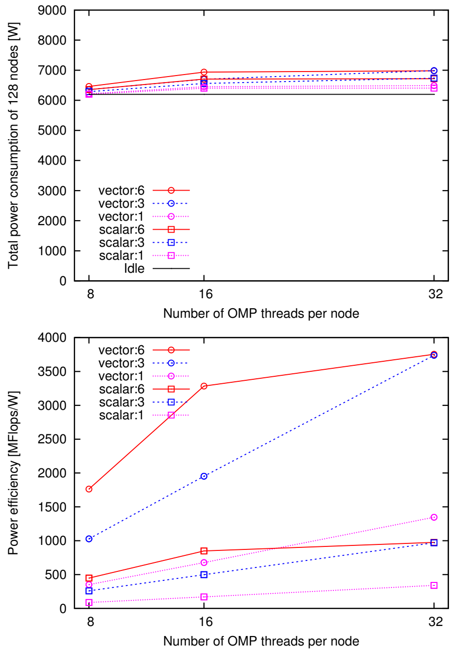

Fig. 4. FPU multiply-add: Power consumption and Power efficiency

QPX vector instructions and standard scalar PowerPC FP instructions, which is executed in one of four slots. QPU employs clock-gating; pruning the clock when not in use.

We observed the power consumption of QPU, running six different synthetic FP stress loops, QPX vector fused multiply-add(FMA) instructions or scalar FMA instructions, with 1, 3 or 6 target registers. Figure 4 shows the power consumption and performance per watt under the FP stress loops on 128 nodes, changing the number of OpenMP threads: 8 (one thread every other core), 16 (one thread per core), and 32 (two threads per core).

The result clearly shows that hardware thread(up to two threads) is power efficient for floating point operations. It also indicates that sub-clock gating may not be implemented for scalar FP operations.

With 32 OpenMP threads with both 3 and 6 target register, the aggregated performance on 128 nodes is almost the theoretical peak $128(nodes) * 1.6(GHz) * 4(vectors) * 2(FMA) => 26.21\,TFlops$. With 16 OpenMP threads, the aggregated performance is 22.79 ,TFlops with 6 registers and 13.08 TFlops with 3 registers.

Unlike BGP PowerPC 450 core, the A2 core is an in-order; the sequence of instructions directly impacts the performance. Most instructions have single cycle throughput, so at the best case, it can issue two instructions from different threads per cycle if one instruction is FP instruction and another instruction is non-FPU(e.g. branch,integer or storage instruction). However, many A2 instructions have a latency of six or seven-cycles. For example, FP instructions have a six-cycle latency and FP load instructons have a seven-cycle latency when it hits in the data cache. If a FP load instruction followed by a FP instruction that uses the target register of the FP load, it incurs a six-cycle penalty. In the FPU benchmark loop ”vector:6” or ”scalar:6”, it uses six target registers, thus no penalty. In ”vector:3” or ”scalar:3”, each instruction incurs a three-cycle penalty. Multi-threading can eliminate those penalty if they can run other threads’ instruction during stall cycles.

Power consumption on memory copying:

We also compared power characteristics on several different memory copying routines (see Figure 5, 6 and **??**). The first is the C library’s memory copy (libc memcpy), which is usually fully optimized for the target hardware. On BGQ, it uses quad load and store instructions and issues prefetching hints if possible. We also tested memory copying with 4-byte and 8-byte operation size, which are common sizes for single-precision and double-precision floating-point data. The buffer size (x axis) is reported per thread, so, for example, in experiments with 32 OpenMP threads per node, or two threads per core, the total memory consumption is double that of 16 OpenMP threads.

The libc memcpy routine outperforms the others if the buffer fits in the L1 or L2 cache. If the buffer is larger, all memory copying routines drops to approx. 0.5 GB/s/W, except the 4-byte copy using 8 threads, which presumably does not have enough parallelism to hide the memory latency.

For main memory access(e.g. 64MB buffer size), our stress code achieved to approximate 26.3 GB/s (per node), which is basically the practical peak performance, although BGQ DDR3 itself aggregated (theoretical) peak performance is 42.6 GB/s with two channels.

For L1 cache access, it achieves 13.7 GB/s (per core) with libc memory, 1.68 GB/s with 4-byte memcpy and 3.34 GB/s with 8-byte memcopy. On BGQ A2, a cache-hit integer load instruction has a five-cycle latency. As previously mentioned, a cache-hit QPU load instruction has a seven-cycle latency. Load instructions execute with one-cycle throughput, thank to a 256-bit load/store interface, so the maximum data rate is 51.2 GB/s with QPU load/store and 12.8 GB/s.

Accuracy of power measurements: Our benchmark applies a constant stress for a fixed and relatively long time period in order to characterize power consumption of primitive operations. In the real world, however, stresses change frequently, and so presumably does consumed power level.

To understand the accuracy of measured power data for

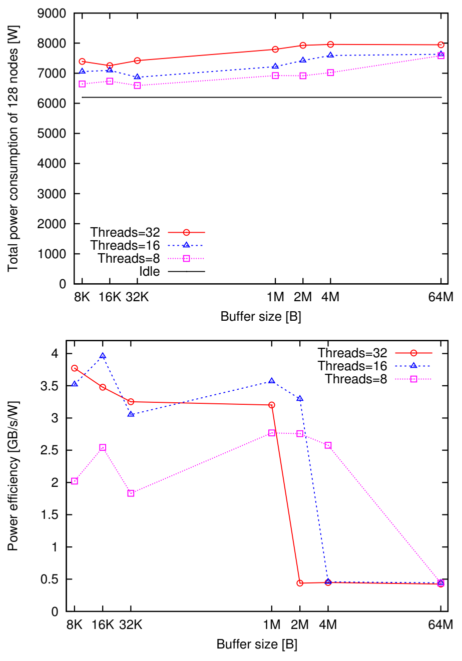

Fig. 5. libc memcopy stress: Power consumption and Power efficiency

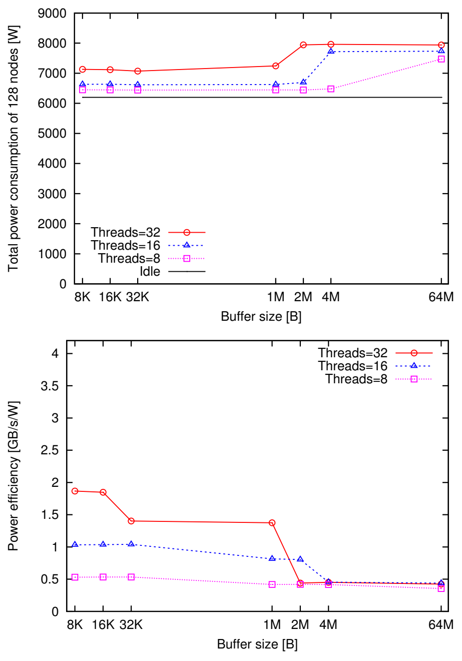

Fig. 7. 8-byte memcopy stress: Power consumption and Power efficiency

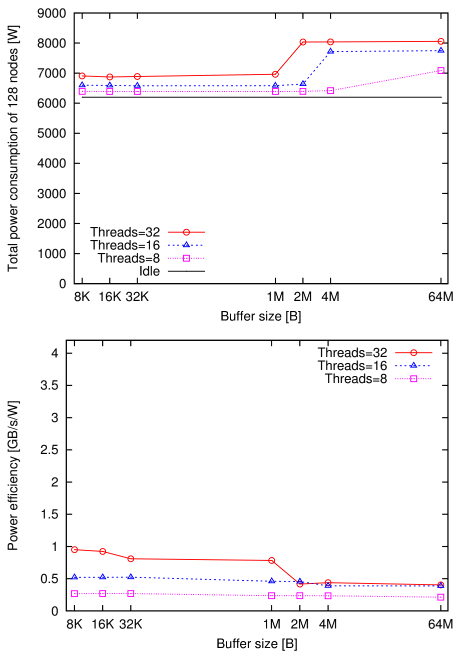

Fig. 6. 4-byte memcopy stress: Power consumption and Power efficiency

short-period activities, we profiled power consumption on a 128-node partition while running intermittent workload patterns that consist of a repeating cycle of stress and sleep. We chose the libc memcpy as the stress. The cycle length was set to 4000ms, with stress period varying from 1ms to 4000ms (100% stress). We estimated the measurement error by calculating the ratio of the average of the measured power consumption and the estimated average power consumption, assuming average consumption is proportional to the stress length.

We observed a 0.04% error on a 1000ms stress and 0.22% error on a 500ms stress. With the duration of the stress smaller than 50ms, the error increases sharply (Figure 8).

At this time, we do not have a clear explanation for what contributes to the error. We suspect a quantization error of the A/D converter to read voltage and current, as well as sampling errors from the internal microcontroller and FPGA.

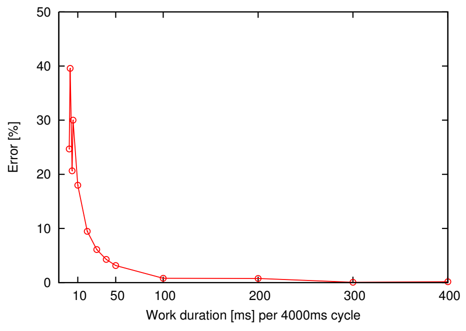

Fig. 8. Power measurement error

Power breakdown: Our profiling code records data on all seven power domains. We compared the breakdown of power consumption on idle (Sleep), FPU (N=16, compiler-generated loop) and memcpy (64 MB buffer per thread) and also a few MPI primitives—see Figure 9.

In our measurements on a 128-node partition, we have not observed any significant change in power consumption on domain 3, 4, 6, and 8 among stresses, including Sleep. As for domain 1 (CPU core), the power consumption on the FPU stress is the highest. The FPU stress consumes an additional 6 W per processor compared with the idle power consumption, while other stresses consume an additional 3 to 5 W, thus overall there is not much difference between them. However, domain 2 (memory) shows more pronounced differences. In particular, the memcpy stress consumes additional 9 W per processor. MPI primitives mostly consume power for memory operations without an increase in domain 1 power, this result indicates that these operations are probably performed by using the DMA engine.

BGQ compute chips employ clock gating. When a component is not in use, the clock to that component is cut off. This approach applies to components such as the FPU; however, other components such as the link chip do not have such feature, so their power consumption is nearly constant regardless of the workload.

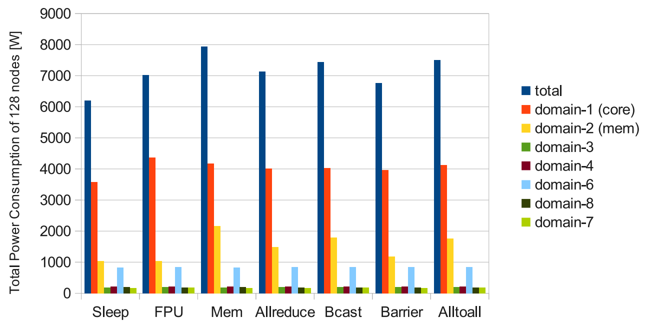

Fig. 9. Breakdown of power consumption

VI. Conclusions

In this paper we evaluated the existing power monitoring capabilities of IBM Blue Gene systems. These capabilities were designed primarily with environmental and health monitoring in mind, but, with the new push toward more energy-efficient computing, we were interested in harnessing them for power consumption measurements of HPC applications.

We found that with a careful choice of benchmarks, it is possible to obtain meaningful power consumption data even on the BGP systems, where the interval of the data stored in the database is measured in minutes. Data so coarsely grained, however, is of little practical use in applications consisting of multiple distinct phases. Also, the cumulative nature of the data from node boards does not aid in identifying the most power consuming subsystems (e.g., CPU vs memory).

In comparison, the new power monitoring capabilities on the BGQ system are a significant improvement. In addition to providing sub-second resolution, we get separate data for CPU, memory, and the interconnect. Crucially, the data is directly accessible to the compute job running on the system. There is still room for improvement, however, such as eliminating the 7-second delay before the data becomes available. With the increasing resolution, the issue of jitter rears its head. Perhaps it would be best if software were to only indicate the beginning and end of a measurement period, with the sampling and integration handled in hardware, so that the software is presented with total energy used rather than an instantaneous power consumption.

With the new monitoring capabilities, we successfully implemented our power-profiling code and demonstrated it on Argonne's early-access BGQ system. We studied power efficiency (performance-per-watt) characteristics on primitive operations; the results show interesting characteristics of BGQ. For example, we observed that the performance-per-watt of vector operations is more than twice that of scalars. This is presumably because there is no subcomponent clock gating in FPU. We also confirmed empirically that two hardware threads per core are better than a single thread for all workloads in the core and L2 cache domain.

We will continue to investigate power monitoring capabilities of emerging HPC systems and will keep developing our power profiling code. Our goal for BGQ is to make the profiling completely transparent (running it on a spare hardware thread of the system core) so that we can profile unmodified application binaries, seeking to improve their power efficiency at leadership-scales.

Acknowledgments

We thank IBM's Paul Coteus and Yutaka Sugawara for in-depth discussion on the power systems on Blue Gene/Q.

We thank IBM's George Chiu, Bryan Rosenburg, Yoonho Park, Heiko Schick, Blake Fitch, Ralph Bellofatto, Robert Germain, Eric Van Hensbergen, James Sexton, Martin Ohmacht, Krishnan Sugavanam, Kyung Ryu, Todd Inglett, and Robert Wisniewski for their advice on the Blue Gene hardware.

We thank Argonne ALCF's William Scullin, Paul Rich, Ray Loy, Jeff Hammond, Kalyanasundaram Kumaran, Vitali Morozov, William Allcock, and Timothy Williams, for their advice on the ALCF early-access Blue Gene/Q machine.

This work was supported by the Office of Advanced Scientific Computer Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357. This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory.

References

[1] DOE: Report from the Architectures and Technology for Extreme Scale Computing Workshop. (12 2009)

[2] Pakbaznia, E., Ghasemazar, M., Pedram, M.: Temperature-aware dynamic resource provisioning in a power-optimized datacenter. In: Proceedings of the Conference on Design, Automation and Test in Europe. DATE '10, 3001 Leuven, Belgium, Belgium, European Design and Automation Association (2010) 124–129

[3] Rasmussen, N.: Calculating total cooling requirements for data centers. American Power Conversion, white paper (2007)

[4] Feng, W., Scogland, T.: The Green500 List: Year one. Parallel and Distributed Processing Symposium, International **0** (2009) 1–7

[5] Feng, W.c., Feng, X., Ge, R.: Green supercomputing comes of age. IT Professional **10** (January 2008) 17–23

[6] Feng, W.c., Cameron, K.: The green500 list: Encouraging sustainable supercomputing. Computer **40** (December 2007) 50–55

[7] Hsu, C.h., Feng, W.c.: A power-aware run-time system for high-performance computing. In: Proceedings of the 2005 ACM/IEEE conference on Supercomputing. SC '05 (2005)

[8] Liu, J., Poff, D., Abali, B.: Evaluating high performance communication: a power perspective. In: Proceedings of the 23rd international conference on Supercomputing. ICS '09, New York, NY, USA, ACM (2009) 326–337

[9] Lynar, T.M., Herbert, R.D., Chivers, S., Chivers, W.J.: Resource allocation to conserve energy in distributed computing. Int. J. Grid Util. Comput. **2**(1) (May 2011) 1–10

[10] Heath, T., Diniz, B., Carrera, E.V., Meira, Jr., W., Bianchini, R.: Energy conservation in heterogeneous server clusters. In: Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming. PPoPP '05, New York, NY, USA, ACM (2005) 186–195

[11] Harada, F., Ushio, T., Nakamoto, Y.: Power-aware resource allocation with fair qos guarantee. In: Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications. RTCSA '06, Washington, DC, USA, IEEE Computer Society (2006) 287–293

[12] Wang, L., von Laszewski, G., Dayal, J., Wang, F.: Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with dvfs. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. CCGRID '10, Washington, DC, USA, IEEE Computer Society (2010) 368–377

[13] hsing Hsu, C., chun Feng, W.: A feasibility analysis of power awareness in commodity-based high-performance clusters. In: Cluster Computing, 2005. IEEE International. (sept. 2005) 1 –10

[14] Garcia, E., Orozco, D., Gao, G.: Energy efficient tiling on a Many-Core Architecture. In: Proceedings of 4th Workshop on Programmability Issues for Heterogeneous Multicores (MULTIPROG-2011); 6th International Conference on High-Performance and Embedded Architectures and Compilers (HiPEAC), Heraklion, Greece (January 2011) 53–66

[15] Feng, X., Ge, R., Cameron, K.W.: Power and energy profiling of scientific applications on distributed systems. Parallel and Distributed Processing Symposium, International **1** (2005) 34

[16] Alam, S., Barrett, R., Bast, M., Fahey, M.R., Kuehn, J., McCurdy, C., Rogers, J., Roth, P., Sankaran, R., Vetter, J.S., Worley, P., Yu, W.: Early evaluation of ibm bluegene/p. In: Proceedings of the 2008 ACM/IEEE conference on Supercomputing. SC '08, Piscataway, NJ, USA, IEEE Press (2008) 23:1–23:12

[17] IBM BG/L Overview. http://www.research.ibm.com/journal/rd/gara.pdf

[18] Top500. http://www.top500.org

[19] IBM: White Paper: The PowerPC 440 Core

[20] Ramanathan, R.: Intel multi-core processors: Making the move to quad-core and beyond. Technology@Intel Magazine (2006)

[21] IBM Blue Gene team: Overview of the IBM Blue Gene/P project. IBM Journal of Research and Development **52**(1/2) (2008) 199–220

[22] Coteus, P., Bickford, H.R., Cipolla, T.M., Crumley, P.G., Gara, A., Hall, S.A., Kopcsay, G.V., Lanzetta, A.P., Mok, L.S., Rand, R., Swetz, R., Takken, T., La Rocca, P., Marroquin, C., Germann, P.R., Jeanson, M.J.: Packaging the blue gene/l supercomputer. IBM J. Res. Dev. **49**(2) (March 2005) 213–248

[23] Haring, R.A., Ohmacht, M., Fox, T.W., Gschwind, M.K., Satterfield, D.L., Sugavanam, K., Coteus, P.W., Heidelberger, P., Blumrich, M.A., Wisniewski, R.W., Gara, A., Chiu, G.L.T., Boyle, P.A., Chist, N.H., Kim, C.: The ibm blue gene/q compute chip. IEEE Micro **32** (2012) 48–60

[24] Argonne Leadership Computing Facility: The ANL Intrepid Blue Gene System. http://www.alcf.anl.gov/intrepid

[25] Gropp, W., Lusk, E., Skjellum, A.: Using MPI: Portable Parallel Programming with the Message Passing Interface. MIT Press (1994)